Elektrobit

Quickstart Guide

EB robinos for EB Assist ADTF Trial

1.1.0, Released

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

EB robinos Support

Phone: +49-9131-7701-8801
Email: support.EB-robinos@elektrobit.com

Legal disclaimer

Confidential and proprietary information.

Table of Contents

1. About this documentation

1.1. Target audience: Developers

Developers use EB robinos for EB Assist ADTF to support the development of Highly Automated Driving (HAD) systems. Since the current EB robinos release does not cover all components of the reference architecture, the documentation focuses on following components:

- EB robinos Grid Fusion, which is used to develop grid based sensor data fusion.
- EB robinos Path Planning, which is used to generate the nearest cost-optimal and obstacle free path from a given start position to a given target position.
- EB robinos Positioning, which is used to provide position information in both a global and a local coordinate system.
- EB robinos Safety Management, which is used to validate data during the execution of the ADTF configuration.

Developers using EB robinos for EB Assist ADTF should have knowledge of:

- Automotive Data and Time-Triggered Framework (ADTF)
- C++ programming

2. Safe and correct use

2.1. Intended usage of EB robinos for EB Assist ADTF

EB robinos for EB Assist ADTF is intended to be used in the development of HAD systems within ADTF.

A special safety concept for your test vehicle must be created, implemented, and assessed according to ISO26262 and your local traffic rules.

2.2. Possible misuse of EB robinos for EB Assist ADTF

EB robinos for EB Assist ADTF version 1.1.0 is not meant to be used in full scale vehicles.

WARNING 	**Possible misuse and liability** You may use the software only as in accordance with the intended usage and as permitted in the applicable license terms and agreements. Elektrobit Automotive GmbH assumes no liability and cannot be held responsible for any use of the software that is not in compliance with the applicable license terms and agreements.

3. Introduction

EB robinos for EB Assist ADTF is a driver assistance platform which has been developed by Elektrobit Automotive GmbH. It is designed to support automotive manufacturers and their suppliers in the development of Highly Automated Driving (HAD) systems. It offers software modules that can be used in the field of autonomous driving.

4. System requirements and prerequisites

EB Assist ADTF knowledge is a prerequisite for using EB robinos for EB Assist ADTF.

EB robinos for EB Assist ADTF has these requirements:

Table 4.1. System requirements

Minimum	**Recommended**
Windows 7 64 bit	
ADTF 2.13.2 64 bit	
7 GB disk space	
4 GB RAM	16 GB RAM
2 x 1 GHz CPU	4 x 2 GHz CPU

EB robinos for EB Assist ADTF is tested and runs in the following environment:

- Microsoft Windows 7 64 Bit
- Microsoft Visual Studio 2010 64 Bit with Service Pack 1 (for developing custom components)
- EB Assist ADTF 2.13.2 for Windows 64 Bit
- EB Assist ADTF Device Toolbox 2.6.1 for Windows 64 Bit
- EB Assist ADTF Display Toolbox 2.1.1 for Windows 64 Bit

5. Installation

The EB robinos for EB Assist ADTF trial can be downloaded directly from the EB robinos website. The software must be activated with a license file. You will receive the software license file by email after you register to download the software.

5.1. Download the software

Open a browser and navigate to the EB robinos webpage: www.eb-robinos.com.

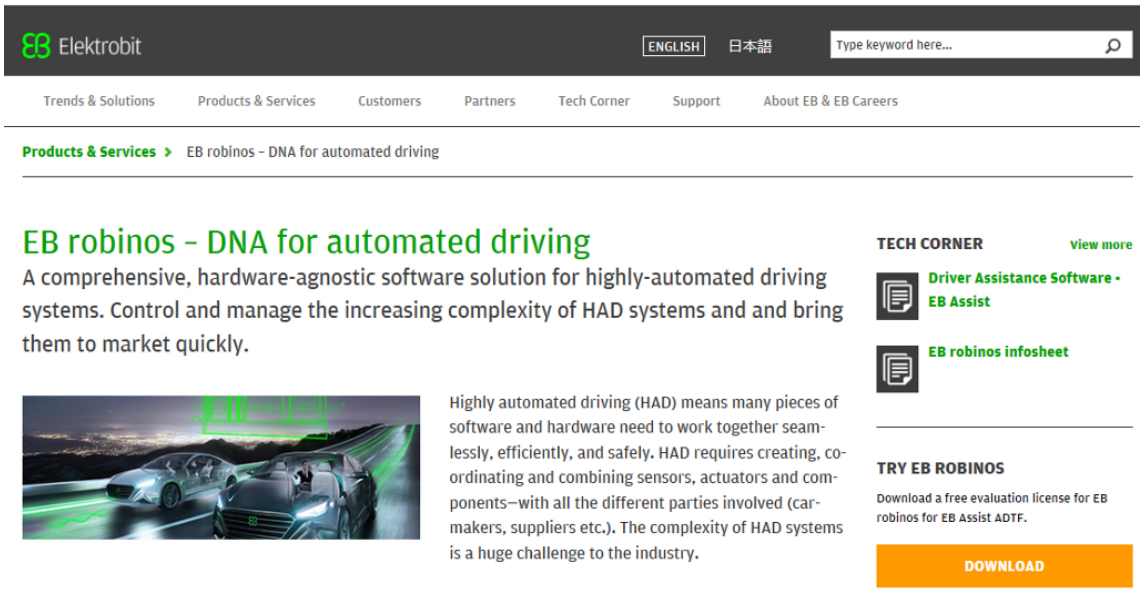


Figure 5.1. The EB robinos webpage

Click the **Download** button under **TRY EB ROBINOS**, or go to www.try-eb-robinos.com.

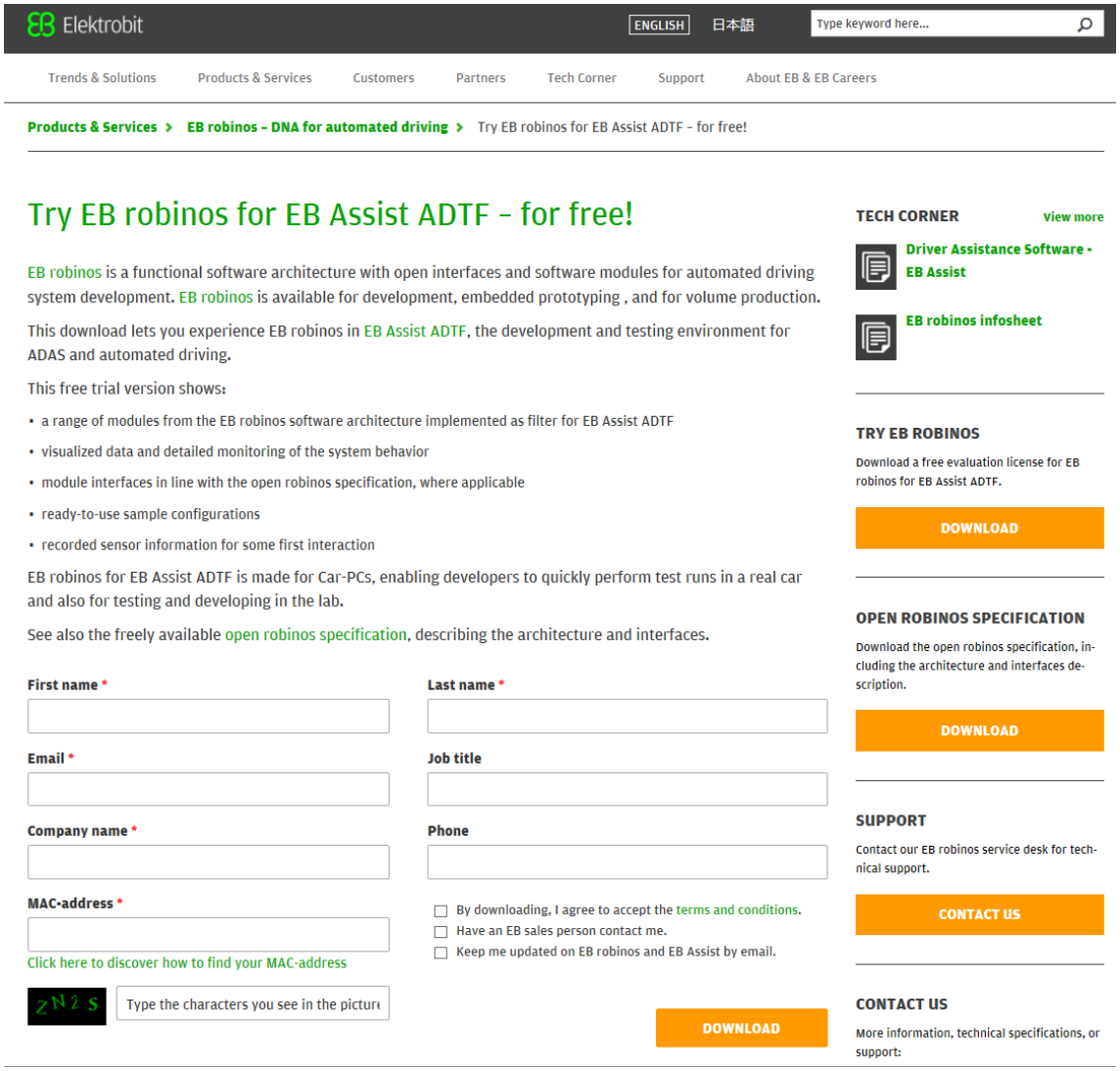


Figure 5.2. EB robinos for EB Assist ADTF license request webpage

Fill in the required information for the trial license request. Mandatory information is indicated by an asterisk.

NOTE

The EB robinos for EB Assist ADTF trial license is sent via email after you register. The license is node-locked. It is linked to the computer MAC-Address and it cannot be used with another computer.

Read and agree to the terms and conditions.

Click the **Download** button and save the EB robinos for EB Assist ADTF trial software (provided as a zip file).

Extract the downloaded zip file and make sure that the following files are available the destination directory:

- `EB_robinos_1.1.0_Trial.exe`

- `EB_robinos_resources.zip`
- `Quickstart_Guide_Trial.pdf`

5.2. Install the software

Double-click on the installer (e.g.: `EB_robinos_1.1.0_Trial.exe`) to start the installation wizard.

Follow the installation wizard to install the software. Make sure that there is enough disk space available for the installation.

Click on **Finish** after the installation is complete.

5.3. Install the example resources

Several example resources are provided in the `EB_robinos_resources.zip` file located in the EB robinos download directory (see Section 5.1).

Extract the contents of `EB_robinos_resources.zip` to the directory where the EB robinos for EB Assist ADTF trial has been installed (e.g.: `C:/EB_robinos_1.1.0`).

5.4. License the software

After the EB robinos for EB Assist ADTF trial software is installed it must be activated with a license file.

If you do not have an evaluation license, it can be requested via the EB robinos webpage www.try-eb-robinos.com (see Section 5.1).

If you do not have an EB Assist ADTF license on the same MAC address you can request one from Elektrobit. Follow the instructions in the email you receive when you register for the EB robinos for EB Assist ADTF trial.

The license files will be sent to you by email. Save the license files in the license directory (e.g.: `C:/EB_robinos_1.1.0/ADTF_2.13.2_64_bit/license/`).

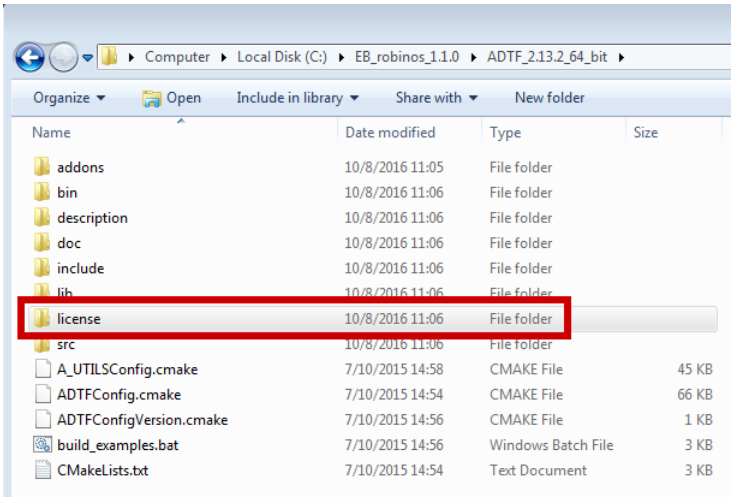


Figure 5.3. EB robinos for EB Assist ADTF trial license

6. Usage

Some short-cut and batch files have been created to simplify the EB robinos examples.

6.1. EB robinos examples

To load an example, go to the EB robinos for EB Assist ADTF trial installation directory (e.g.: `C:/EB_robinos_1.1.0`) and start the corresponding batch file for the desired demo:

- `Start_EB_robinos_DS_Grid_Fusion_Demo.bat`
- `Start_EB_robinos_Bayes_Grid_Fusion_Demo.bat`
- `Start_EB_robinos_Height_Grid_Fusion_Demo.bat`
- `Start_EB_robinos_Path_Planning_Demo.bat`
- `Start_EB_robinos_Positioning_Demo.bat`
- `Start_EB_robinos_Safety_Management_Demo.bat`
- `Start_EB_robinos_Valet_Parking_Demo.bat`

NOTE

The first time you launch the EB robinos for EB Assist ADTF trial, you may see a message on the EB Assist ADTF splash screen stating that the license is invalid. The message will only be shown if you do not have an existing EB Assist ADTF license. See section 5.4, "License the software"

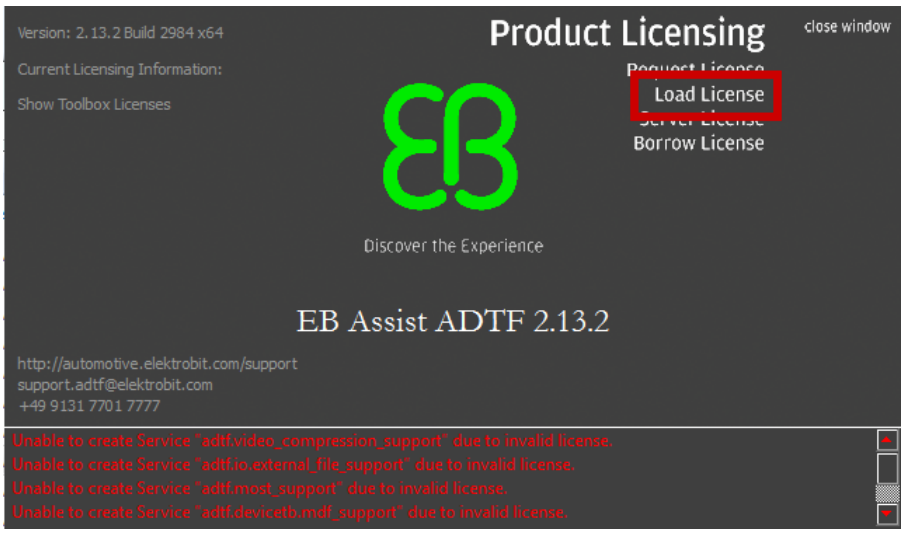


Figure 6.1. EB Assist ADTF Product Licensing message

Click **Load License** and browse to the license directory (e.g.: `C:/EB_robinos_1.1.0/ADTF_2.13.2_64_bit/license/`).

Select the EB Assist ADTF license (do not select the EB robinos license) and click **OK**.

After the project is loaded, it can be started with the sidebar buttons. Click the **Play** button and the EB Assist ADTF configuration is executed.

Figure 6.2. Click the play button

6.1.1. EB robinos Grid Fusion Dempster-Shafer and Bayes examples

This section provides two prepared examples which show the Grid Fusion capabilities, in terms of occupancy information, when it models the environment around an Ego. The required steps to load them are presented in section 6.1.1.1. The only difference between the examples is the mathematical approach behind the implementation of each one. The first example uses Dempster-Shafer theory while the second example uses Bayes

theory. Since the scenario used is the same, there is a common detailed description of the examples that can be found in section 6.1.1.2.

6.1.1.1. Load an EB robinos Grid Fusion occupancy example

To load an EB robinos Grid Fusion occupancy example, go to the installation directory and start one of the corresponding batch files:

- `Start_EB_robinos_DS_Grid_Fusion_Demo.bat`
- `Start_EB_robinos_Bayes_Grid_Fusion_Demo.bat`

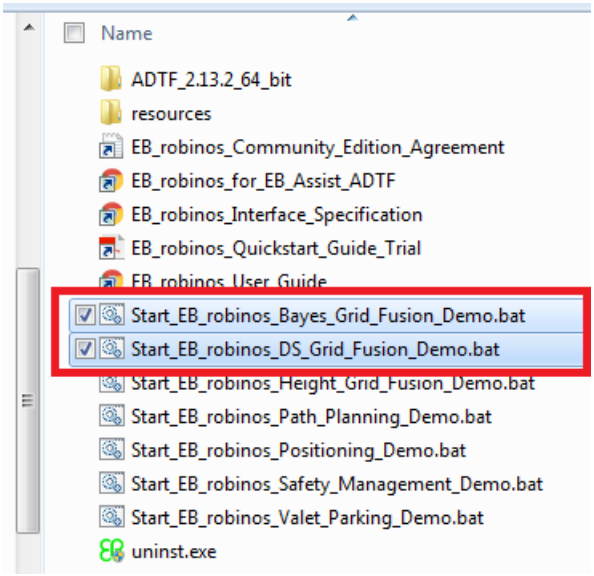


Figure 6.3. Load a Grid Fusion occupancy example

After the project is loaded, it can be started with the sidebar buttons. Click the **Play** button and the EB Assist ADTF configuration is executed.

Figure 6.4. Click the Play button

6.1.1.2. EB robinos Grid Fusion occupancy example

In this example, an EB Assist ADTF demo configuration for EB robinos Grid Fusion is implemented that shows the Grid Fusion capabilities, in terms of occupancy information, when it models the environment around an Ego.

It uses an aerial image representing an inner city map area. In this area, an earth fixed grid with the size 80x80 [m] is selected (all properties are already set). There is an ego car that is driving inner city (its icon is already placed in the configuration). The ego car is equipped with two IBEO Lidar sensors, one sensor on the front, one on the rear. In this example, measurements from the IBEO Lidar sensor have been recorded in real conditions. Using the *Harddisk_Player* from ADTF device toolbox, the recorded raw data is processed offline with the Grid Fusion.

The raw data provided by sensors are processed with two Lidar sensor models. For each sensor model a maximum distance range of 50 meters is set. The measurements are processed only if they are placed inside of the earth fixed grid. The software always shifts the grid in such a way that it includes the moving car.

After the sensors' measurements are processed, the results are stored in the grid layer. An application specific view of the grid data is used in order to visualize the occupied and free areas (see figure 6.5). All drivable areas are marked with green. The display shows all valid Lidar measurements, which are interpreted as relevant obstacle objects, in red. The grid layer stores the actual data. The content of the grid layer is degrading over time, but it is updated over and over again by the sensors' measurements.

Figure 6.5. Snapshot from the running configuration

A helper window that displays information coming from the front video camera is shown in figure 6.6.

Figure 6.6. Front camera snapshot

The common controls are used to play/pause the loaded configuration (see figure 6.7) or to set the speed of the playback with a time factor (see figure 6.8).

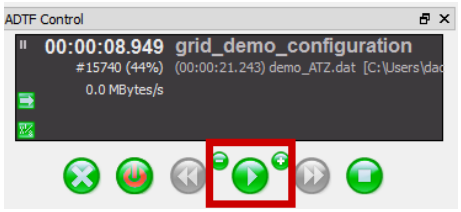


Figure 6.7. Play/pause button

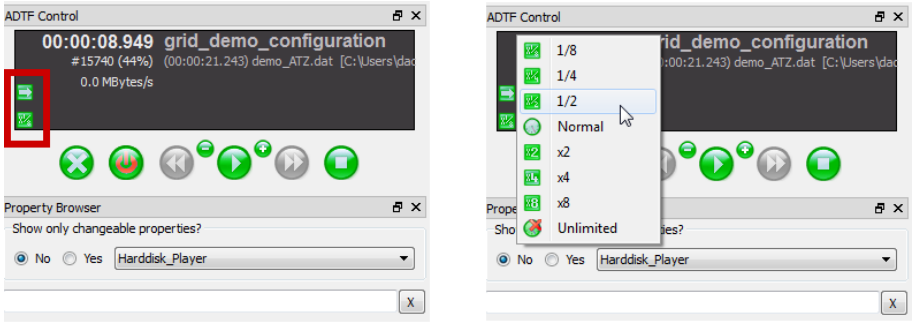


Figure 6.8. Set the playback speed

6.1.2. EB robinos Grid Fusion height map example

This section provides a prepared example which shows the Grid Fusion capabilities, in terms of height information, when it models the environment around an Ego. The required steps to load it are presented in section 6.1.2.1. A detailed description of the example can be found in section 6.1.2.2.

6.1.2.1. Load the EB robinos Grid Fusion height map example

To load the EB robinos Grid Fusion height map example, go to the installation directory and start the following batch file:

- `Start_EB_robinos_Height_Grid_Fusion_Demo.bat`

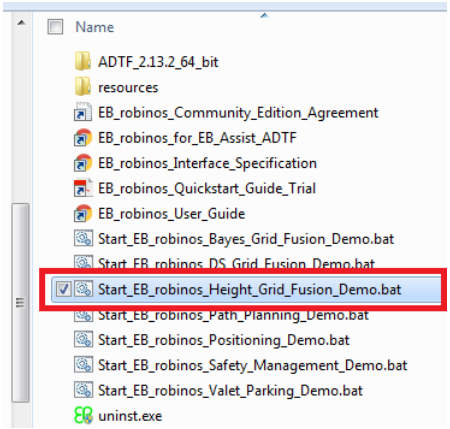


Figure 6.9. Load the Grid Fusion height map example

After the project is loaded, it can be started with the sidebar buttons. Click the **Play** button and the EB Assist ADTF configuration is executed.

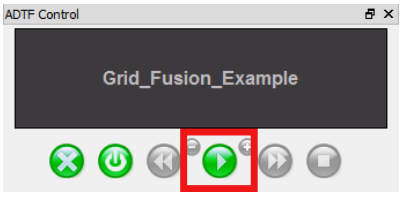


Figure 6.10. Click the Play button

6.1.2.2. EB robinos Grid Fusion height map example

In this example, an EB Assist ADTF demo configuration for EB robinos Grid Fusion is implemented that shows the Grid Fusion capabilities, in terms of height information, when it models the environment around an Ego.

It uses an aerial image representing an inner-city map area. In this area, two earth-fixed grids with the size 48 x 48 [m] are selected (all properties are already set): one grid is able to store height information and another one is able to store occupancy information. The grids are pointing to the same area. The height grid could have been set independently, but using it together with an occupancy grid helps provide a better understanding of the location where the height information of the obstacles were detected.

There is an Ego car that is driving in an inner city (its icon is already placed in the configuration). The Ego car is equipped with a Hokuyo UST-10LX sensor placed in the front of the car. The sensor looks at the ground at an angle of about 77 degrees. The data provided by the sensor is converted from a 2D-line into 3D-point cloud data by doing a 2D-3D-mapping based on an extrinsic calibration.

In this example, measurements from the Hokuyo sensor have been recorded in real conditions. Using the Harddisk_Player from the ADTF device toolbox, the recorded raw data is processed offline with the Grid Fusion.

The raw data provided by the sensor is processed with two SfM sensor models: one height sensor model for updating the height grid and one occupancy sensor model for updating the occupancy grid. The two grids are updated in parallel and use the same input data provided by the Hokuyo sensor.

For each sensor model the maximum distance range is set to 20 meters. The measurements are processed only if they are placed inside of the earth-fixed grid. The software always shifts the grids in such a way that it includes the moving car.

After the sensor's measurements are processed, the results are stored in the height grid layer, and the respective occupancy grid layer.

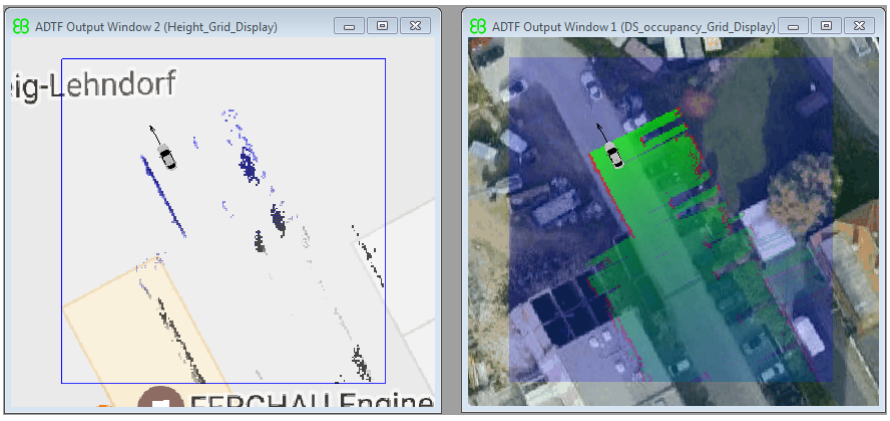


Figure 6.11. Snapshot from the running configuration

An application specific view of the grid data is used for the occupancy grid layer in order to visualize the occupied and free areas (see figure 6.11 right side).

All drivable areas are marked with green. The display shows all valid SfM measurements, which are interpreted as relevant obstacle objects, in red. The occupancy grid layer stores the actual occupancy data. The content of the occupancy grid layer degrades over time (by decreasing the intensity of green and red colors), but it is updated over and over again by the sensor's measurements.

An application specific view of the grid data is used for the height grid layer in order to visualize the height of detected objects (see figure 6.11 left side).

The smallest entity of the grid represents a grid cell, therefore the height information is displayed at cell level. Each object covers one or more cells in the grid. Gray-scale colors are used to specify the height of the objects where darker gray indicates higher objects.

There is a value that expresses a confidence in that height information for the height information represented at cell level. Conventionally, the confidence information is represented as a blue rectangular mark inside a cell. The larger the rectangle the higher the confidence.

The height layer stores the actual height data and the confidence in the height data. The content of the height grid layer degrades over time (by decreasing the confidence information for each cell), but it is updated over and over again by the sensor's measurements. The example in figure 6.12 shows that an object was detected to the left side of the car. Next to the car, the cells confidence is high, and behind the car the confidence becomes smaller and smaller because these cells are not updated anymore.

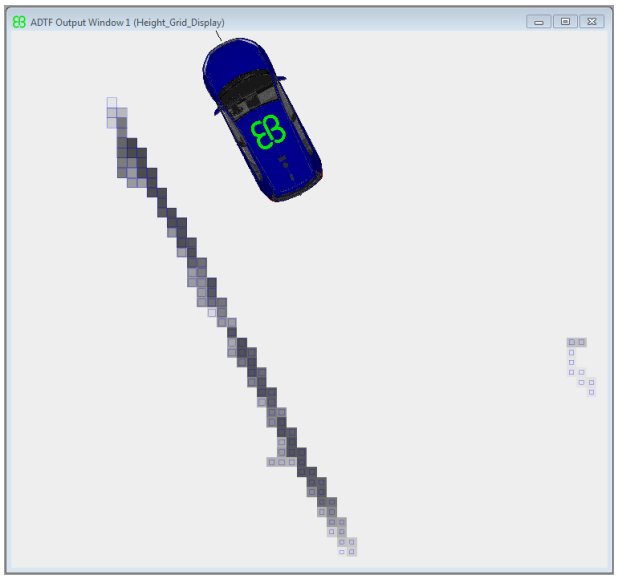


Figure 6.12. Example of height layer degrading process

The common controls are used to play/pause the loaded configuration (see figure 6.13) or to set the speed of the playback with a time factor (see figure 6.14).

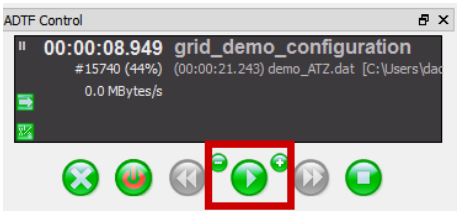


Figure 6.13. Play/pause button

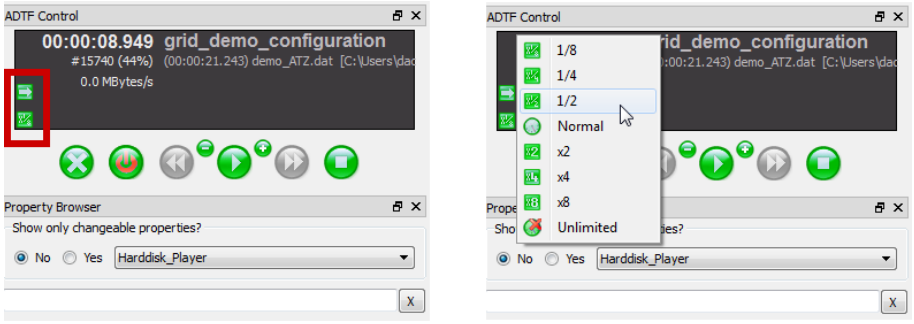


Figure 6.14. Set the playback speed

6.1.3. EB robinos Path Planning example

This example demonstrates EB robinos Path Planning in a residential area in the inner city. The real life recording contains Ego data, front and rear Lidar sensors, and a camera. It was recorded with a real driver and no EB robinos Path Planning was used at recording time. The EB robinos Path Planning will plan a more efficient route that does not perfectly match the path followed by the real driver. The camera in this example is just for documentation purposes. The two Lidar sensors and the road boundaries, which are generated by OSM data, are merged in a grid layer from the EB robinos grid fusion. Based on this grid information the EB robinos Path Planning searches the best path to a regularly updated target position.

The main parts of the EB robinos Path Planning configuration are: seven PP Data Source filters, five PP Algorithm filters, and a scheduler.

The PP Data Source filters EB robinos PP Configuration Space and EB robinos PP Node Based Data are used to provide the grid data to the EB robinos Path Planning algorithms. The EB robinos PP Cost Parameters provides a list of properties to individual costs for specific tasks like a gear change. The EB robinos PP Precomputed Costs ensures that at any position with respect to the target position, the minimal costs for reaching the target position without considering obstacles is known. The EB robinos PP Turning Radius provides a suitable maximal turning radius depending on the current velocity. The EB robinos PP OSM Graph generates an OSM graph which can be accessed by the algorithms. The EB robinos PP Data Source Resetter creates synchronous updates of all connected EB robinos PP Data Sources and the connected external data sources.

The PP algorithm filters use the information from the PP Data Source filters. A detailed description of EB robinos Path Planning algorithm filters can be found in the EB robinos for EB Assist ADTF User Guide:

- EB robinos PP Last Path Adapter
- EB robinos PP Graph Based Path Planning
- EB robinos PP Reeds Shepp Path Model
- EB robinos PP Hybrid A Star Filter
- EB robinos PP Rapidly Exploring Random Tree

The EB robinos PP Scheduler executes all connected algorithm assemblies and chooses the best generated path. It also defines whether a recalculation of the path must be performed or a path update is sufficient. A recalculation is forced after a defined time, when the distance between the vehicle and the path exceeds a defined threshold, or when there are obstacles in the given path that must be avoided. Finally, the path description is modified by the EB robinos PP Velocity Planning to plan the velocity for each control point in the path.

With the EB robinos PP Path View Subfilter the path description can be visualized in a 2D Display. This is illustrated in Figure 6.15.

Figure 6.15. Screenshot of the EB robinos Path Planning visualization

6.1.4. EB robinos Positioning example

This section provides one prepared example that shows the EB robinos Positioning capabilities.

6.1.4.1. Load the EB robinos Positioning example

To load the EB robinos Positioning example, go to the installation directory and start the batch file:

- `Start_EB_robinos_Positioning_Demo.bat`

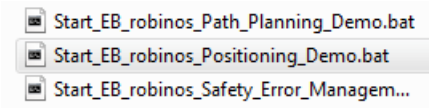


Figure 6.16. Load a Positioning example

After the project is loaded, it can be started with the sidebar buttons. Click the **Play** button and the EB Assist ADTF configuration is executed.

Figure 6.17. Click the Play button

6.1.4.2. EB robinos Positioning example

In this example, an EB Assist ADTF demo configuration for EB robinos Positioning is implemented. It uses an aerial image representing a mapped area, a video stream, and a signal view displaying height information (see figure 6.18). There are three Ego vehicles that approach the Nuremberg (Germany) airport at the beginning of the recording, drive through a parking garage, and then continue towards an urban area (their icons are already placed in the configuration). The gray vehicle represents the reference position that is coming from an iMar iTrace. The red vehicle represents the fused local position of the Eb robinos Positioning which is embedded into the global coordinate system with a delay of 10 seconds. Due to the embedding, the local position will show up 10 seconds after the start of the demo. The green vehicle represents the fused position that is provided by the EB robinos Positioning. The EB robinos Positioning uses uncalibrated gyro, accelerometer, and raw GPS sensor information from the iTrace device. In addition, odometer and reverse information from the vehicle's CAN bus are used to enhance the position. The test data has been recorded from a real test drive and is processed with a Harddisk_Player during playback.

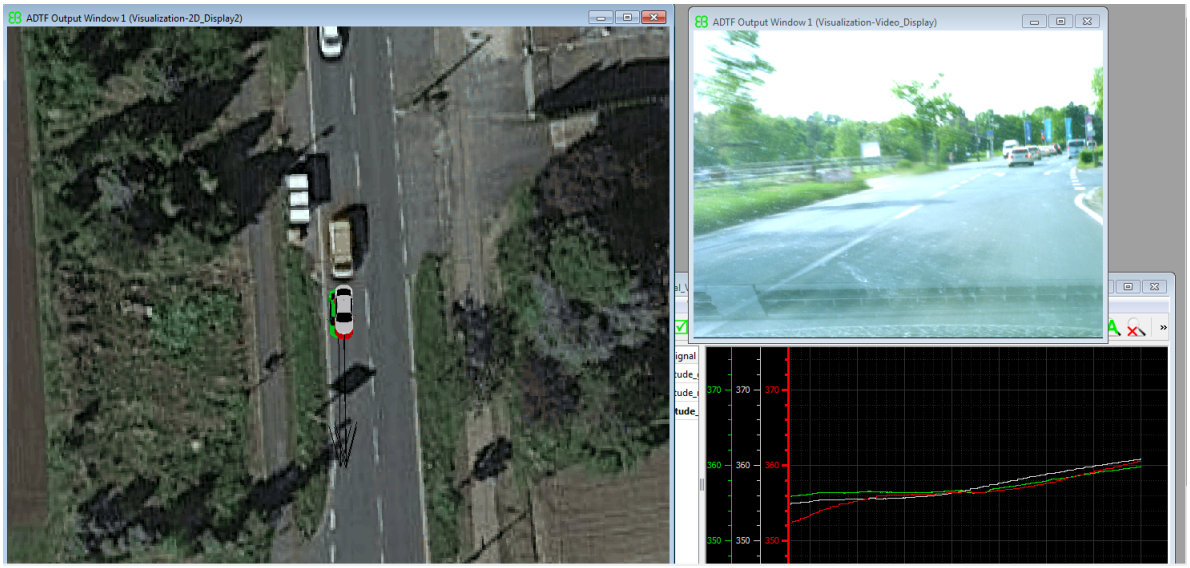


Figure 6.18. Snapshot from the running configuration

The common controls are used to play/pause the loaded configuration (see figure 6.19) or to set the speed of the playback with a time factor (see figure 6.20).

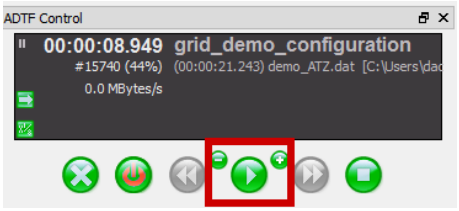


Figure 6.19. Play/pause button

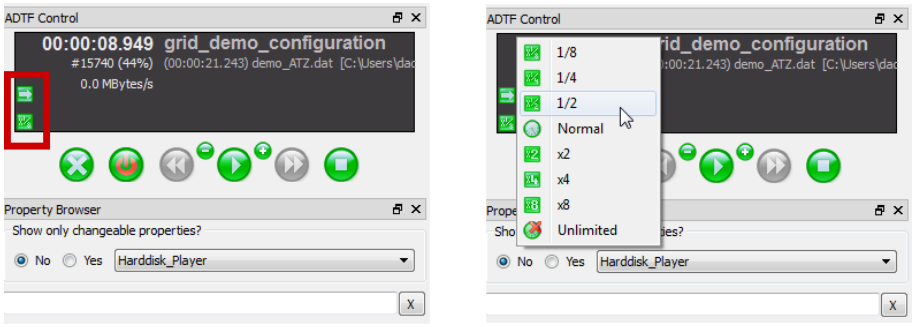


Figure 6.20. Set the playback speed

6.1.5. EB robinos SM example

The EB robinos SM example demonstrates error validation in a valet parking scenario.

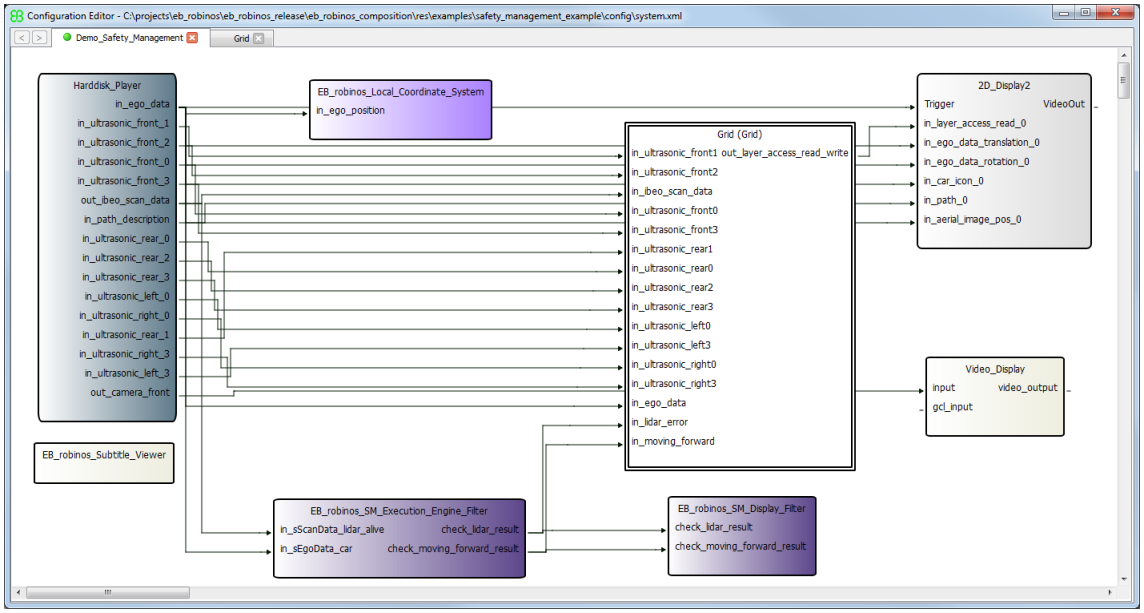


Figure 6.21. EB robinos SM Example ADTF configuration

The SM example shown in figure 6.21 focuses on the data validation of the scenario.

The provided Lidar data stream is prepared with interruptions which are tracked by the Safety Management. The second validity check triggers, when the car drives backwards.

The EB robinos SM Execution Engine filter receives Lidar data and the Ego data of the car to execute the defined validation tests on the data. The results are transmitted by the result pins for each defined error strategy. The result pins are connected to the SM Display filter which shows the current error state of all checked conditions in a traffic light interface figure 6.22. Other software components can react to a triggered error condition. The cycle time of the validation can be changed with a property in the EB robinos SM Execution Engine filter.

The SM Execution Engine filter was compiled with the generated source code that was specified in the SM Error Strategy Editor. The configuration used for the SM Error Strategy Editor is included in this example.

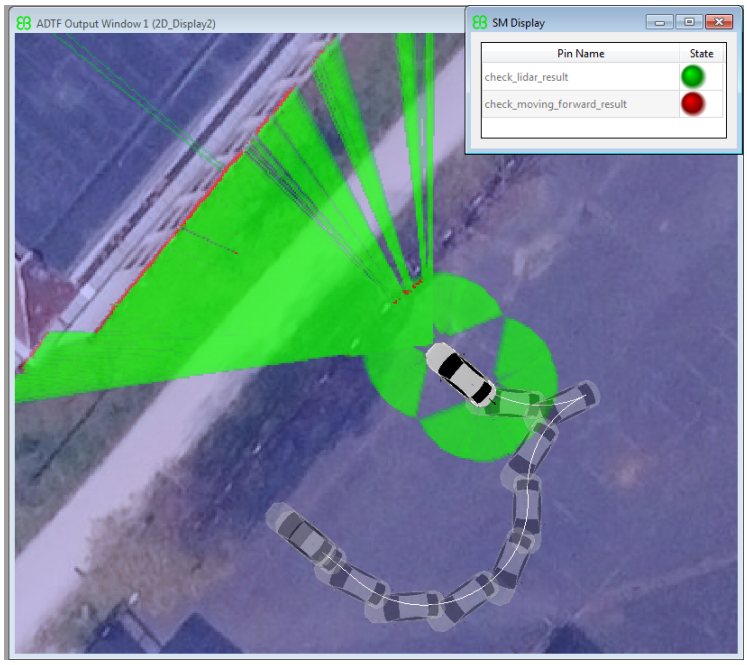


Figure 6.22. EB robinos SM Example ADTF validation

6.1.6. EB robinos valet parking example

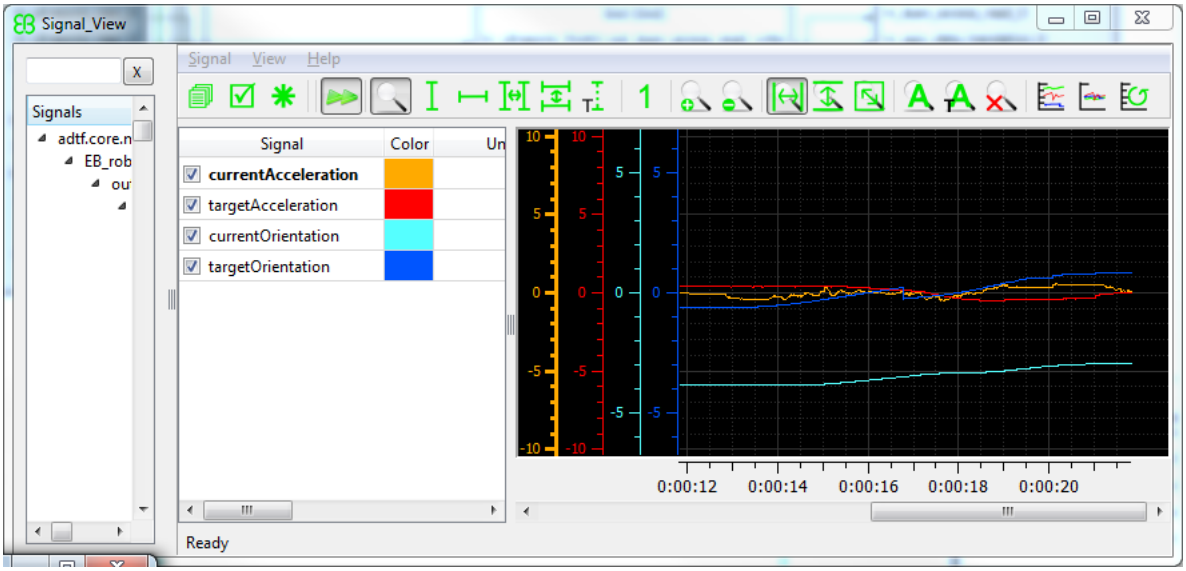
In this example the valet parking scenario is demonstrated. Triggered by a remote input signal, in this case a mobile application, the vehicle autonomously drives from its current position to a defined target position.

Based on the grid, which holds environment information, the path to the target position is planned by the EB robinos Path Planning and then executed by the vehicle controller. The real life recording provides the sensor information, the planned path description, and the vehicle ego information. For a closer look at the EB robinos Path Planning and Grid Fusion refer to the examples in section 6.1.1.2 and section 6.1.3.

In this recording the vehicle is blocked by an obstacle. First, the vehicle reverses and turns away from the obstacle. The vehicle then performs a gear change and drives forward, steering as required until it reaches the target position. This is illustrated in figure 6.23.

Figure 6.23. Screenshot of valet parking scenario while reversing

While performing the parking maneuver, control parameters such as steering and acceleration can be monitored at run time using the **Signal View** in the EB Assist ADTF display toolbox. This is illustrated in figure 6.24. For more information about the **Signal view** please consult the EB Assist ADTF display toolbox documentation.



Current acceleration is shown in orange, target acceleration is shown in red.
Current orientation is shown in light blue, target orientation is shown in blue.

Figure 6.24. EB Assist ADTF Signal View, monitoring vehicle control parameters

6.2. Control the 2D-Display

Use the standard Windows controls to resize the 2D-Display windows.

Use the mouse wheel or the +/- keys to change the zoom in the active 2D-Display window.

Click and drag to change the view position in the 2D-Display.

To reset the 2D-Display view, right click inside the window and select **Reset View** from the context menu. The 2D-Display view is reset to a car-centered view at the default zoom.

7. Reference information

For more information, refer to the EB robinos for EB Assist ADTF User Guide. Additional information and the latest news about EB robinos for EB Assist ADTF can be found on the Elektrobit webpage (www.eb-robinos.com).

Appendix A. Potential export restrictions

The product is not classified as “Dual-Use Items” in accordance with Council Regulation (EC) 428/2009 (as of June 2016). The License is responsible for compliance with any applicable export or import regulations or obligations. Elektrobit Automotive GmbH provides assistance on request.

Appendix B. Open source libraries

EB robinos for EB Assist ADTF uses code from several open source libraries.

Boost
: EB robinos for EB Assist ADTF uses code from Boost 1.61, licensed under the Boost Software License 1.-0. The code is used without any modifications. The library sources are available from the Boost website, or can be obtained by sending a request to support.EB-robinos@elektrobit.com.

Flann
: EB robinos for EB Assist ADTF uses code from Flann 1.8.4, licensed under the 2-Clause BSD License. The code is used without any modifications. The library sources are available from the Flann website, or can be obtained by sending a request to support.EB-robinos@elektrobit.com.

googlemock
: EB robinos for EB Assist ADTF uses code from googlemock 1.7.0, licensed under the 2-Clause BSD License. The code is used without any modifications. The library sources are available from the Google Test website, or can be obtained by sending a request to support.EB-robinos@elektrobit.com.

Proj
: EB robinos for EB Assist ADTF uses code from Proj 4.9.1, licensed under an MIT License. The code is used without any modifications. The library sources are available from the Proj4 website, or can be obtained by sending a request to support.EB-robinos@elektrobit.com.

openCV
: EB robinos for EB Assist ADTF uses code from openCV 2.4.11, licensed under the 3-Clause BSD License. The code is used without any modifications. The library sources are available from the openCV website, or can be obtained by sending a request to support.EB-robinos@elektrobit.com.

QT
: EB robinos for EB Assist ADTF uses code from QT 4.7.1, licensed under LGPL 2.1. The code is used without any modifications. The library sources are available from the QT website, or can be obtained by sending a request to support.EB-robinos@elektrobit.com.